Cheatography

MongoDB Shell Cheat Sheet

by Volodymyr Nerovnia (nerv) via cheatography.com/152613/cs/32856/

Examples connection strings

mongosh "mongodb://localhost:27017"
mongosh "mongodb://mongodb0.example.com:28015" --username alice --authenticationDatabase admin
mongosh "mongodb+srv://server.example.com/"
mongosh "mongodb://mongodb0.example.com:28015/?tls=true"
mongosh "mongodb://localhost:27017/db1"
mongosh "mongodb://mongodb0.example.com:28015" --tls
mongosh "mongodb://mongodb0.example.com:28015/?tls=true"

Use an Editor

config.set("editor", "vi")	using an external editor, set the editor from within mongosh
edit	start a new editing session
var albums = [ ]; edit albums	edit a variable
edit db.collection.insertMany([])	edit a statement
.editor	start the built-in editor
export EDITOR=vi	Set the EDITOR environment variable in bash or zsh
process.env.EDITOR = 'nano'	set environment variables from within mongosh
export EDITOR="/usr/local/bin/code --wait"	using Visual Studio as an external editor; set an environment variable
config.set("editor", "code --wait")	using Visual Studio as an external editor
config.set("editor", null) or process.env.EDITOR = ''	unset the external editor
<ctrl> + d	exit and run your function
<ctrl> + c	exit without running your function

Methods

db.<collection>.insertOne(<object>);	insert document to collection
db.<collection>.insertMany()	insert multiple documents
db.<collection_name>.find({<filter_expression>})	query documents in a collection
pretty()	get pretty view information
count()	get count documents
explaine()	

Methods (cont)

{ <field>: { "$elemMatch": { <field>: <value> } } }	elemMatch matches document that contain abn array field with at least one element that matches the specified query criteria.
{ <field1>: { <operator1>: <value1> }, ... }	form of query operators in a query filter document
db.<collection_name>.findOne({<filter_expression>})	find one document in a collection
db.<collection>.find({<query>}, {<projection>})	1 - include the field 0 - exclude the field
it	iterates through a cursor; cursor - a pointer to a result set of a query; pointer is a direct address of the memory location
db.<collection_name>.insertOne(<object>)	insert one document
db.<collection_name>.insert([<collection objects>], { "ordered": true/false })	insert documents
db.<collection_name>.insertMany([<array_objects>])	insert documents
"ordered": false	insert all records thay are not duplicated
"ordered": true	insert only records before duplucate, other will not insert
db.collection.updateOne()	Updates a single document within the collection based on the filter.
db.collection.updateMany()	Updates all documents that match the specified filter for a collection.
db.collection.findAndModify()	Modifies and returns a single document.
db.collection.findOneAndUpdate()	Updates a single document based on the filter and sort criteria.
db.collection.findOneAndReplace()	Replaces a single document based on the specified filter.
db.collection.bulkWrite()	Performs multiple write operations with controls for order of execution.

Aggregation Structure

db.collection.aggregate ([{ stage1 }, { stage2 }, { ...stageN }], {options})

$fieldName	field path expression
$$SYSTEMVARIABLE	system level variable
$$UserVariable	user variable

Comparison Query Selectors

$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$in	Matches any of the values specified in an array.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$nin	Matches none of the values specified in an array.

Logical Query Selectors

$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not	Inverts the effect of a query expression and returns documents that do not match the query expression.
$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

Element Query Selectors

$exists	Matches documents that have the specified field.
$type	Selects documents if a field is of the specified type.

Evaluation Query Selectors

$expr	Allows use of aggregation expressions within the query language.
$jsonSchema	Validate documents against the given JSON Schema.
$mod	Performs a modulo operation on the value of a field and selects documents with a specified result.
$regex	Selects documents where values match a specified regular expression.
$text	Performs text search.
$where	Matches documents that satisfy a JavaScript expression.

Geospatial Query Selectors

$geoIntersects	Selects geometries that intersect with a GeoJSON geometry. The 2dsphere index supports $geoIntersects.
$geoWithin	Selects geometries within a bounding GeoJSON geometry. The 2dsphere and 2d indexes support $geoWithin.
$near	Returns geospatial objects in proximity to a point. Requires a geospatial index. The 2dsphere and 2d indexes support $near.
$nearSphere	Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The 2dsphere and 2d indexes support $nearSphere.

Array Query Selectors

$all	Matches arrays that contain all elements specified in the query.
$elemMatch	Selects documents if element in the array field matches all the specified $elemMatch conditions.
$size	Selects documents if the array field is a specified size.

Bitwise Query Selectors

$bitsAllClear	Matches numeric or binary values in which a set of bit positions all have a value of 0.
$bitsAllSet	Matches numeric or binary values in which a set of bit positions all have a value of 1.

Bitwise Query Selectors (cont)

$bitsAnyClear	Matches numeric or binary values in which any bit from a set of bit positions has a value of 0.
$bitsAnySet	Matches numeric or binary values in which any bit from a set of bit positions has a value of 1.

Projection Operators

$	Projects the first element in an array that matches the query condition.
$elemMatch	Projects the first element in an array that matches the specified $elemMatch condition.
$meta	Projects the document's score assigned during $text operation.
$slice	Limits the number of elements projected from an array. Supports skip and limit slices.

Miscellaneous Operators

$comment	Adds a comment to a query predicate.
$rand	Generates a random float between 0 and 1.

System Variables

NOW	the current datetime value
CLUSTER_TIME	the current timestamp value, only available on replica sets and sharded clusters
ROOT	References the root document
CURRENT	References the start of the field path being processed in the aggregation pipeline stage
REMOVE	A variable which evaluates to the missing value.
DESCEND	One of the allowed results of a $redact expression.
PRUNE	One of the allowed results of a $redact expression.
KEEP	One of the allowed results of a $redact expression.

Aggregation Pipeline Stages

$let	Binds variables for use in the specified expression, and returns the result of the expression.
$redact	Restricts the contents of the documents based on information stored in the documents themselves.

Aggregation Pipeline Stages (cont)

$map	Applies an expression to each item in an array and returns an array with the applied results.
$abs	Returns the absolute value of a number.
$accumulator	Defines a custom accumulator operator
$acos	Returns the inverse cosine (arc cosine) of a value.
$acosh	Returns the inverse hyperbolic cosine (hyperbolic arc cosine) of a value.
$add	Adds numbers together or adds numbers and a date.
$addToSet	returns an array of all unique values that results from applying an expression to each document in a group.
$allElementsTrue	Evaluates an array as a set and returns true if no element in the array is false
$and	Evaluates one or more expressions and returns true if all of the expressions are true or if run with no argument expressions.
$anyElementTrue	Evaluates an array as a set and returns true if any of the elements are true and false otherwise
$arrayElemAt	Returns the element at the specified array index.
$arrayToObject	Converts an array into a single document; the array must be either:
$asin	Returns the inverse sine (arc sine) of a value.
$asinh	Returns the inverse hyperbolic sine (hyperbolic arc sine) of a value.
$atan	Returns the inverse tangent (arc tangent) of a value.
$atan2	Returns the inverse tangent
$atanh	Returns the inverse hyperbolic tangent
$avg	Returns the average value of the numeric values.
$binarySize	Returns the size of a given string or binary data value's content in bytes.

Creating a user

```
use admin
db.createUser({
  user: "m103-admin",
  pwd: "m103-pass",
  roles: [
    {role: "root", db: "admin"}
  ]
})
```

Service methods & shell commands

db.getMongo()	verify your current database connection
db.status()	get database status
db.getSiblingDB()	get access to a different database from the current database without switching your current database context
db	display the database you are using
show dbs	get list of databases
use <database>	switch databases or create a new database
show collections	get list of collections
use admin; db.shutdownServer()	another example shutdown server
db.shutdownServer({timeoutSecs : 5})	shutdown with timeout
db.adminCommand({shutdown : 1, timeoutSecs : 5})	with timeout
db.adminCommand({shutdown : 1, force : true})	force replica set shutdown; use If there is no up-to-date secondary and you want the primary to shut down
.exit, exit, or exit()	exit from shell
quit or quit()	exit from shell
Ctrl + C twice	exit from shell

Replica Set Commands

rs.initiate()	initiating the Replica Set
rs.status()	getting Replica Set status
db.serverStatus()['repl']	

Replica Set Commands (cont)

rs.printReplicationInfo()	Get current oplog data (including first and last event times, and configured oplog size)
rs.add(<name>)	adding other members to Replica Set
rs.addArb(<name>)	adding arbiter to Replica Set
rs.isMaster()	getting an overview of the Replica Set topology
rs.conf()	get current configuration
rs.reconfig(<cfg_var>)	reconfigure Replica Set
rs.remove()	remove Replica Set
rs.stepDown()	stepping down the current primary
db.oplog.rs.find()	query the oplog after connected to a replica set
db.oplog.rs.stats().capped	verify that this collection is capped
db.oplog.rs.stats().size	get current size of the oplog
db.oplog.rs.stats().maxSize	get size limit of the oplog
rs.slaveOk()	enabling read commands on a secondary node
sh.startBalancer(timeout, interval)	start the balancer
sh.stopBalancer(timeout, interval)	stop the balancer
sh.getBalancerState()	see if the balancer is enabled
sh.setBalancerState(boolean)	enable/disable the balancer
sh.isBalancerRunning()	check if balancer is running

Sharding Commands

mongos -f mongos.conf	start the mongos server
sh.status()	check sharding status
sh.addShard("m103-repl/192.168.103.100:27012")	
db.products.createIndex(<key>)	create an index
sh.shardCollection(<collection>, <key>)	shard a collection
sh.enableSharding("m103")	

Good Shard Key

Non-monotonic change
High Cardinality
Low Frequency

mongod service

mongod	start MongoDB not as service, using all defaults
sudo service mongodb status	get status
sudo service mongodb start	start service
sudo service mongodb stop	stop service
sudo service mongodb restart	restart service
kill <mongod process ID> or kill -2 <mongod process ID>	kill mongod process
sudo lsof -i -P -n	get list open ports
sudo netstat -tulpn | grep LISTEN	get list open ports
tail-f <log_file>	read log file

mongod arguments

--host	uri host
--port	port
--username	user name
--authenticationDatabase	name authentication database
--tls	use tls
--dbpath	path to database
--repair	repaire database
-f <path_to config_file>	use not default configuration file
--shutdown	shutdown server
--fork	start mongod as daemon
--logpath	set path to log file

Compressors

zlib	supported
zstd	not supported
snappy	not supported

Config Replica Set

```
storage:
  dbPath: /var/mongodb/db/node1
net:
  bindIp: 192.168.103.100,localhost
```

Config Replica Set (cont)

```
  port: 27011
security:
  authorization: enabled
  keyFile: /var/mongodb/pki/m103-keyfile
systemLog:
  destination: file
  path: /var/mongodb/db/node1/mongod.log
  logAppend: true
processManagement:
  fork: true
replication:
  replSetName: m103-example
```

Creating the keyfile

```
sudo mkdir -p /var/mongodb/pki/
sudo chown vagrant:vagrant /var/mongodb/pki/
openssl rand -base64 741 > /var/mongodb/pki/m103-
keyfile
chmod 400 /var/mongodb/pki/m103-keyfile
```

Backup & Restore

mongoexport --uri="mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies" --collection=sales --out=sales.json	backup JSON
mongodump --forceTableScan --uri mongodb+srv://<user_name>:<password>@<host_name>/<database>	backup BSON
mongodump --uri "mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies"	backup BSON
mongoimport --uri="mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies" --drop sales.json	restore JSON
mongorestore --uri="mongodb://<user_name>:<password>@<host_name>/<database>" dump/	restore BSON
mongorestore --uri "mongodb+srv://<your username>:<your password>@<your cluster>.mongodb.net/sample_supplies" --drop dump	restore BSON

Mongo Keyboard Shortcuts

Up-arrow	previous-history
Down-arrow	next-history
Home	beginning-of-line
End	end-of-line
Tab	autocomplete
Left-arrow	backward-character
Right-arrow	forward-character
Ctrl-left-arrow	backward-word
Ctrl-right-arrow	forward-word
Meta-left-arrow	backward-word
Meta-right-arrow	forward-word

Mongo Keyboard Shortcuts (cont)

Ctrl-A	beginning-of-line
Ctrl-B	backward-char
Ctrl-C	exit-shell
Ctrl-D	delete-char (or exit shell)
Ctrl-E	end-of-line
Ctrl-F	forward-char
Ctrl-G	abort
Ctrl-J	accept-line
Ctrl-K	kill-line
Ctrl-L	clear-screen
Ctrl-M	accept-line
Ctrl-N	next-history
Ctrl-P	previous-history
Ctrl-R	reverse-search-history
Ctrl-S	forward-search-history
Ctrl-T	transpose-chars
Ctrl-U	unix-line-discard
Ctrl-W	unix-word-rubout
Ctrl-Y	yank
Ctrl-Z	Suspend (job control works in linux)
Ctrl-H (i.e. Backspace)	backward-delete-char
Ctrl-I (i.e. Tab)	complete
Meta-B	backward-word
Meta-C	capitalize-word
Meta-D	kill-word
Meta-F	forward-word
Meta-L	downcase-word
Meta-U	upcase-word
Meta-Y	yank-pop
Meta-[Backspace]	backward-kill-word
Meta-<	beginning-of-history
Meta->	end-of-history